Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 589 – 595

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Secured Temporal Log Management Techniques for Cloud

S. Muthurajkumar[a,*], S.Ganapathy[b], M. Vijayalakshmi[a], A. Kannan[a]

[a]*Department of Information Science and Technology, College of Engineering Guindy, Anna University, Chennai, India*
[b]*Department of Computer Applications, Maulana Azad National Institude of Technology, Bhopal, MP, India.*

Abstract

Log Management has been an important service in Cloud Computing. In any business, maintaining the log records securely over a particular period of time is absolutely necessary for various reasons such as auditing, forensic analysis, evidence etc. In this work, Integrity and confidentiality of the log records are maintained at every stage of Log Management namely the Log Generation phase, Transmission phase and Storage phase. In addition to this, Log records may often contain sensitive information about the organization which should not be leaked to the outside world. In this paper, Temporal Secured Cloud Log Management Algorithm techniques are implemented to provide security to maintain transaction history in cloud within time period. In this work, security to temporal log management is provided by encrypting the log data before they are stored in the cloud storage. They are also stored in batches for easy retrieval. This work was implemented in Java programming language in the Google drive environment.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

Keywords: Cloud computing; Log management; Cloud Storage; Cloud Security;

1. Introduction

Logs are machine or transaction data generated by any sort of application or the infrastructure used to run that application. Logs in cloud environment consist details. Such as user identifier, system identifier, operation done and

* Corresponding Author: Tel: +91-94448-82247;
E-mail address: muthurajkumarss@gmail.com

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.098

time. In addition, some logs provide security based settings including access control and usage history.In order to analysis the user behavior, it is necessary to find the anomalous behaviour of user with respect to their usage pattern. In the field of database system, logs are maintained to perform effective recovery of transactions. In such a scenario, they use two types of logs namely undo logs and redo logs.

There are many security flaws in the existing recovery protocols which make them vulnerable during the transmission of log messages and it doesn't have any security mechanism to authenticate the data received by the syslog daemon. Sometimes the log records need to be exposed to the external auditors for secure auditing of logs. So the log data are to be maintained temporally before it can be rotated or deleted in order to maintain the history. These log information are used for recovering data from transaction failures.

The Cloud based logging provides low cost solution for organizations. The main problem in cloud storage is data integrity and confidentiality. In this work, these problems are handled by providing encryption methods to provide correctness, tamper resistance, confidentiality and integrity.

In this work, new techniques for log security in cloud are designed and implemented. For this purpose, the log data are aggregated to form batches and are encrypted and stored. The main advantage of encrypting log files is that the user behavior analysis cannot be carried out by unauthorized users.

2. Literature Survey

There are many works in the literature that discuss about new techniques for log creation, maintenance and analysis. In addition, different research works have been carried out on security in storage retrieval and communication. The related works discuss about the maintenance of forward integrity attack models, key sharing techniques and log file security.

Ma and G.Tsudik[1] describe the need for forward secure stream integrity. They proposed a scheme for secure logging through a new technique known as "Forward-Secure Sequential Aggregate" which is used for authentication. The main advantage of their work is the improvement in security.

Schneierand Kelsey[2] proposed a logging scheme that ensures that the attacker is denied access. Bellare and Yee[3] described the importance of forward integrity and its application in the field of intrusion detection and accountability. The notion of forward integrity is achieved by the use of message authentication codes. In their scheme, even if the attacker gains control of the system, he could only erase the entries. He could never modify or create new entries in the log file.

Holt[4] provides strong cryptographic assurances that the data stored by a logging facility before a system compromise, cannot be modified after the compromise, without detection. He describes how the process of log creation can be separated from log verification process. The major improvements in security and concurrency.

Dolev and Yao[5] proposed a system or a model that could detect the vulnerabilities of an improperly designed protocol. They argued a protocol which is weak in design can be compromised easily during transit. They proposed several models as well as characterizations and algorithms to determine the protocol security. Their scheme encodes the details on sender, plain and cipher texts.

In the key sharing techniques proposed by Shamir[6] attacking entity may sequentially compromise every node and obtain the k pieces. In such a case, the attacker gets all the pieces of the key and can use the pieces to generate the original key. This spoils the effectiveness of the scheme.

Blakely[7] also proposed a method for safeguarding cryptographic keys. He argues that certain cryptographic keys such as the system master key and several other keys in DES cryptosystem present a dilemma. If too many copies are distributed one might go missing or if few keys are made, they might all get destroyed.

Jeong and Kwon[8] et al. proposed a scheme for securing Diffie algorithm to provide session security using keys. Key independence refers to a stronger notion of security and means that session keys are computed independent of each other.

Indrajit Ray[9] et al. described the algorithms for log file preparation, upload tag generation and the algorithms necessary for uploading the log data to the cloud and also the algorithms for rotating and deleting log records. They described a novel scheme for transforming and transmitting the log records to the cloud storage. Computing with all the schemes present in the literature the scheme proposed in this paper is different since it provides a new encryption based security methods for secured log maintenance.

3. System Architecture

The architecture of the system developed in this work consists of various components that include the log generators, log monitor, logging client and logging cloud. A log generator connects with logging client through wire or wireless network and logging client, log monitor connects directly through wireless or wired network.The client connects to the cloud storage through an anonymous channel or secure channel depending upon the situation. Fig. 3. shows the overall system architecture of system.

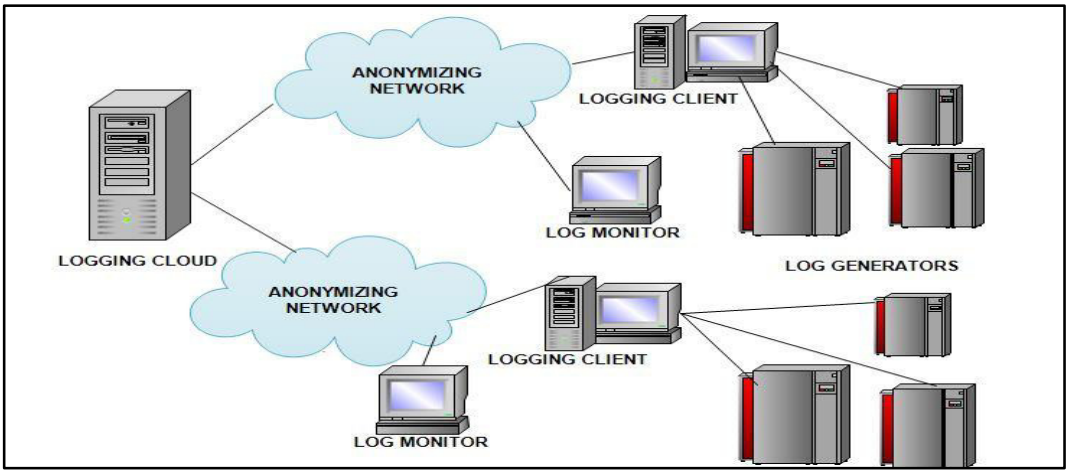

Fig. 1. System architecture.

Logging client consists servers and databases for storing data effectively. The clients can store their data efficiently in the server using this proposed system.

The logging cloud acts as a massive storage medium and also provides many features like anytime-anywhere accessibility, enhanced data security, faster deployment and low cost opportunities. It also provides maintenance service over an extended period of time to log data. Only those who have paid and subscribed to the cloud storage services can upload the data. Having many business organizations as customers greatly reduces the operating cost of the service provider. Log Monitor methods to monitor the log information.

In this work, log analysis helps to enhance the data and transaction security.

4. Temporal Secured Cloud Log Management Algorithm

The proposed Temporal Secured Cloud Log Management Algorithm contains four modules namely log file preparation, anonymous upload tag generation, anonymous upload and anonymous deletion and rotation.

4.1 Log File Preparation

Before the log file is pushed to the cloud for storage, it is encrypted and message authentication codes are appended to the log records to ensure forward integrity. Three keys are used in this work, two for integrity and one for confidentiality. The first entry consists of the timestamp, special log-initialization symbol and the batch size. The last entry is the log close entry which denotes the end of the batch. The steps of this algorithm are adopted based on [1] as follows:

Step1: Generate three master keys A0 and X0 for integrity and K0 for confidentiality.

Step2: Generate the first log entry L0 =(Time_Stamp, init_log, n).Calculate MAC0= HA0[EK0[L0]] .

Step3: Re-compute keys A1 =H[A0], X1 =H[X0] and K1 =H[K0].Delete A0 , X0, K0.

Step4: Create a record M1=L1 || HA0[EK0[L0]. Calculate MAC1 =HA1[EK1[M1].calculate aggregated MAC'1= HnX1[MAC0||MAC1||n].

Step5: Generate a Log close entry LC= [EKn+1[Time_Stamp, log-close || MACn] ,HAn+1[Ekn+1[Time_Stamp, log-close || MACn]]. Upload the batch.

At the end of this algorithm, the log file is ready to be uploaded to the cloud. This algorithm executes in logging client module. The raw log records which are accumulated from the log generators are processed using this algorithm and then transformed into secured sealed log file.

4.2 Anonymous Upload Tag Generation

In order to retrieve the log batch, uploaded to the cloud, before, it is necessary to index the log batches by unique key value. However, the upload tags should be such that it cannot be traced back to the user. In order to generate such a tag, Diffie Hellmen key generation process is used in this work and then hashed to obtain the tag.The algorithm executes in log client and monitor modules. The steps of this algorithm are as follows:

Step1:The log data are stored in the cloud and are indexed using auto-generated tags.

Step2: Choose a generating element α and a primitive root p in a group.

Step3: The logging client A and monitor B generates a random numbers rA and rB keeps it secret.

Step4: A calculates TA=SigA [αrA mod p] and B calculates TB= SigB [αrB mod p]. Exchange TA and TB

Step5: Calculate Diffie-Hellmen Key as T=αrA αrB mod p

Step6: Create the i-th upload-tag, by hashing the key (m−i) used in Diffie Hellmen algorithm times. i.e. upload_tagi = H(m−i) [αrA αrB mod p].

Each upload tag is associated with a log group. To retrieve the log group a can query the cloud using the upload tag.

4.3 Anonymous Upload

After the upload tag is generated, it is associated with a log batch, which consists of encrypted log records. Every log batch is associated with a delete-tag which is generated by using a strong cryptographic hash function of a pseudo random number. This algorithm executes in logging client. The steps of the algorithm are as follows:

Step1: Create a delete_Tag=Hn[randk(·)] using a strong cryptographic function for every log batch.

Step2: The delete_Tag is encrypted using the public key of the cloud using DT=EpubCloud[DeleteTag].

Step3:LogData is sent using the parameters <Upload−Tag, Time_Stamp, DT, N, Loginfo> through an anonymous channel.

After the generation of delete tag, the log group is stored in the cloud with the corresponding tags group.

4.4 Anonymous Deletion and Rotation

This algorithm is used to delete and rotate logs in the cloud storage securely. Deleting and rotating of log records can be done only through authorized personnel only. So the entity which requests to delete must provide a proof, which says the entity has the necessary authorization to delete or rotate. The steps of the algorithm are given below:

Step1:Obtain the relevant deleteTag and UploadTag.

Step2: Check whether the request satisfies the tag given by Tag =Upload− tag ∧ Delete_Tag = DeleteTag.

Step3: Give the proof of authorization as Hn−1[randk(·)]

Step4: The cloud check whether DeleteTag = H[Hn-1 [randk(·)]]. If it is equal, them it deletes or rotates the log accordingly.

5. Performance Analysis

This work has been implemented in Eucalyptus cloud. The results obtained from the experimental are explained using tables in this paper. Table 1. shows the comparison on the number of requests denied by Secured Cloud Log Management Algorithm (SCLMA) and Temporal Secured Cloud Log Management Algorithm (TSCLMA) in 5

experiments with different number of client request. The client requests included genuine and malicious user request with a proportion 18:2. From Table1, it can be observed that the proposed Temporal Secure Cloud Log Management Algorithm model performs better when compared with Secure Cloud Log Management model in restricting the users and provides more than 92% detection and prevention accuracy. This is due to the use of intelligent agents and effective key sharing techniques proposed and used in this model.

Table 1. Number of user requests denied access by SCLMA and TSCLMA

Exp. No	No. of User Request Tried	No. of requests denied by SCLMA	No. of requests denied by TSCLMA
Exp1	100	5	7
Exp2	200	8	11
Exp3	300	12	15
Exp4	400	15	19
Exp5	500	16	24

Fig. 2. shows the security level of number of authorized users who were permitted by the Temporal Secured Cloud Log Management Algorithm model. From this figure, it is observed that the access permission of Temporal Secured Cloud Log Management Algorithm is more than Secured Cloud Log Management Algorithm model. Moreover, 5% of less users where denied access in comparison with the existing system and hence the security is enhanced. This is due to the fact that temporal constraints are used effectively to check the abnormal users.

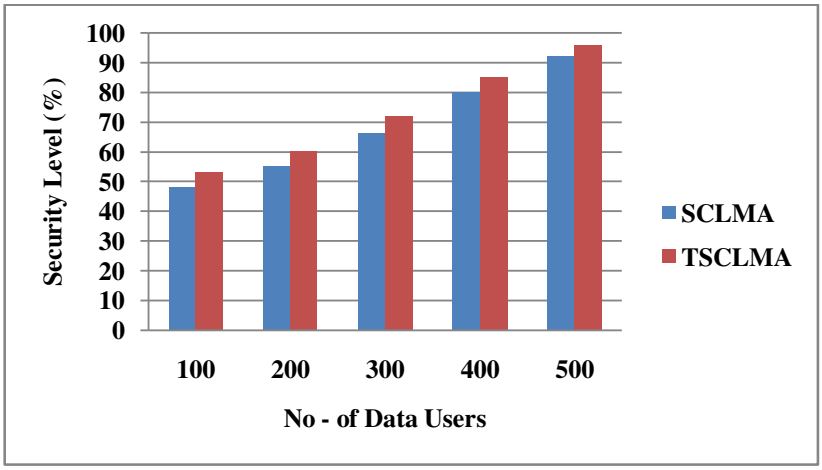

Fig. 2. The number of data centers security level.

Table 2. shows the number of user's requests denied access by the Secured Cloud Log Management Algorithm model after checking identity and Temporal Secured Cloud Log Management Algorithm model to access the database.

Table 2. The number of user request access by temporal cloud log management algorithm

Exp. No	No. of User Request Tried	No. of requests denied by Secured Cloud Log Management Algorithm	No. of requests denied by Temporal Secured Cloud Log Management Algorithm
Exp1	100	7	8
Exp2	200	9	13
Exp3	300	12	20
Exp4	400	17	27
Exp5	500	21	34

Fig. 3. shows the memory requirements comparison for Secured Cloud Log Management Model and Temporal Secured Cloud Log Management Model during time interval (t1, t2). From the implementation carried out in this model, it is observed that there is a difference of 7% memory is used in the proposed model due to the compression made encryption.

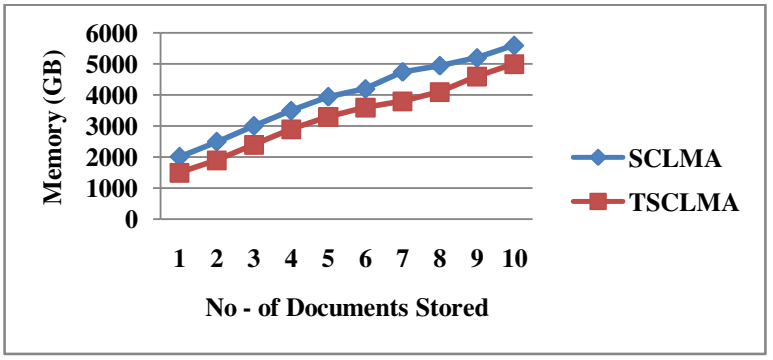

Fig. 3. Memory analysis for static document.

Fig. 4. shows the memory requirements comparison for Secured Cloud Log Management Model and Temporal Secured Cloud Log Management Model during time interval (t1, t2). From the implementation carried out in this model, it is observed that there is a difference of 10% memory is used in the proposed model due to the compression made encryption. However, the memory taken to process is less than compared with Secured Cloud Log Management Algorithm and Temporal Secured Cloud Log Management Algorithm model.

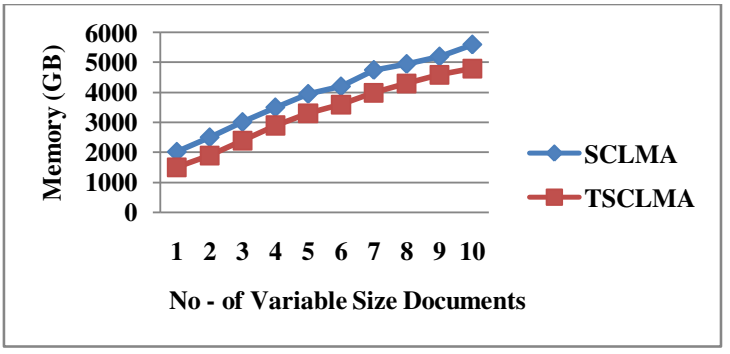

Fig. 4. Memory analysis for dynamic documents.

Fig. 5. shows, the number of violation take to revoke permissions when they are query requested by the user. From the graph, it can be seen that the memory required for processing revoke is proportional to the number of user sessions and number of users who requested for rollback at a time. Even though the revoking memory increases with the number of users, it decreases after certain limit due to the application logic and presence of intelligent agent who learn the behavior. Therefore, the proposed Temporal Secured Cloud Log Management Algorithm consumes less revoking permission when compared to Secured Cloud Log Management Algorithm model.

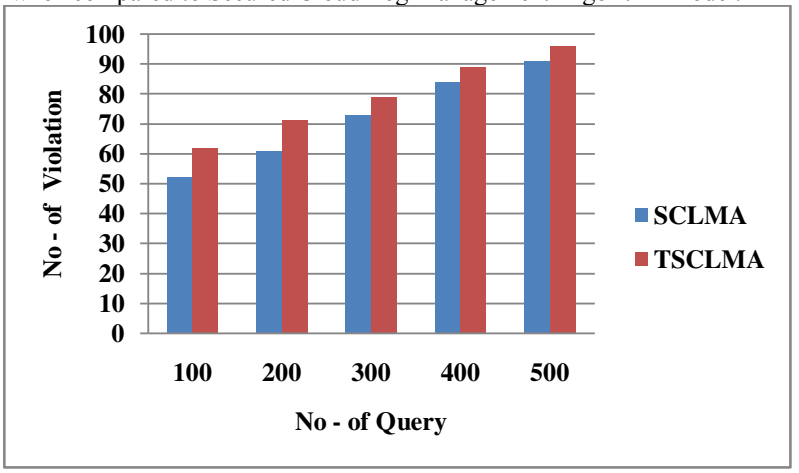

Fig. 5. Revoking permission.

6. Conclusion

In recent times, the size of log files are growing rapidly. Storage and maintenance of large volume of log data locally involves huge investment and significant hardware resources. Cloud storage provides lower startup costs, increased data security, continuous availability, reduced costs, faster deployment and a highly scalable and customizable interface. In this work, we developed a system which used suitable techniques for securely storing log records to data storage. This paper discussed the existing log management protocols and identified the vulnerabilities. The then used a new log management technique that addresses not only security and performance is proposed. Finally, it is proved that the implemented secure logging techniques that sends the generated log records securely to the data storage. In the future, work planned to integrate the log management process directly to the system, rather having to rely on third party libraries and protocols for generation and transmission of logs. In addition to plan to investigate further on to the secure key management techniques.

References

1. Ma D, Tsudik G. A new approach to secure logging. *ACM transactions storage* 2009; **5**,1:2:1–2:21.
2. Schneier B, Kelsey J. Security audit logs to support computer forensics. *ACM transactions on information system security* 1999;**2**,2: 159–176.
3. Bellare M, Yee BS. Forward integrity for secure audit logs. Department of computer science, University of California, San Diego, Technical Reports, November 1997.
4. HoltJ. E. Logcrypt: Forward security and public verification for secure audit logs. *In proceedings of the 4th australasian information security workshop* 2006; 203–211.
5. Dolev D, Yao Y. On the security of public key protocols. *IEEE transactions on information theory* 1983;**29**:2,198–208.
6. Shamir A. How to share a secret. *ACM journal* 1979;**22**,11:612–613.
7. Blakley G.R. Safeguarding cryptographic keys. *In Proceedings of the National Computer Conference* 1979; 313-314.
8. Rae Jeong, Jeong Ok Kwon, Dong Hoon Le. Strong Diffie-Hellman-DSA key exchange. *IEEE Communiacations Letters* 2007;**11**,5:432-433.
9. Ray I, Belyaev K, Strizhov M, Mulamba D, Rajaram M. Secure logging as a service—delegating log management to the cloud. *Systems Journal, IEEE* 2013;**7**,2:323-334.